Ficha Prática 5

Python e Bases de Dados

Bases de Dados

Departamento de Engenharia Informática

1 2 ⚜ 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Objetivos:

- Conhecer o funcionamento de instruções e operações básicas em Python, nomeadamente instruções de controlo condicional, instruções de repetição, e manipulação de strings

Requisitos:

- Python 3.7.4
- Módulo *psycopg2 2.8.3*
- Editor PyCharm
- Tabelas *emp* e *dep* criadas e preenchidas com os dados fornecidos em *scripts-sql.zip*

Material de Apoio:

- Slides t6

1. O seguinte código permite obter todos os dados da tabela *emp*, presente na base de dados. Examine o código em detalhe, pois será a base dos exercícios seguintes.

```
import psycopg2

# A função connect permite estabelecer uma ligação a uma base de dados
# Verifique se a password é igual à que escolheu na instalação de PostgreSQL
conn = psycopg2.connect("host=localhost dbname=postgres user=postgres password=postgres")


# Cria um objecto (cursor) que permite executar operações sobre a base de dados
cur = conn.cursor()

# Efectua uma consulta à base de dados
cur.execute("SELECT * FROM emp;")

# mostra todos os resultados
print(cur.fetchall())

# Fecha a ligação à base de dados
cur.close()
conn.close()
```

Note que, em programas simples, a criação de uma ligação à base de dados deve ser efetuada apenas uma vez (devendo ser fechada antes do programa terminar).
Execute agora o código e examine o resultado detalhadamente. Após executar o código escreva *help(cur)* no interpretador e examine as funções disponibilizadas por este objecto *cursor*.
Altere o código para mostrar a informação presente na tabela dos departamentos.

2. A linha de código *print(cur.fetchall())* imprime uma lista de resultados. Cada uma das linhas resultante da consulta designa-se por tuplo (um conjunto de valores separados por vírgulas). É possível apresentar o resultado de uma consulta de uma forma mais legível. Para isso podemos substituir *print(cur.fetchall())* por:

```
for linha in cur.fetchall():
    print(linha)
```

Execute e verifique que a impressão dos resultados é agora feita de forma diferente. O que ficou diferente?

É possível ainda desempacotar cada linha resultante da consulta em variáveis. Para isto basta atribuir cada linha obtida a X variáveis (X deve corresponder ao número de colunas devolvido pela consulta). Por exemplo, considerando a solução que implementou em resposta à primeira pergunta, o seguinte código irá apresentar a informação dos departamentos de forma mais legível:

```
for linha in cur.fetchall():
    ndep, nome, local = linha
    print(ndep, nome, local)
```

Altere agora a *query* SQL para obter todos os departamentos existentes em localidades terminadas com 'a'. Modifique ainda o código para mostrar o resultado na seguinte forma:

```
Nome: Contabilidade Localidade: Condeixa
Nome: Investigacao Localidade: Mealhada
Nome: Vendas Localidade: Coimbra
```

3. Substitua o ciclo *for* (e instruções associadas), utilizado na pergunta anterior, por apenas:

```
print(cur.fetchone())
```

O que acontece quando executa o programa? Adicione mais uma linha igual, junto à que acabou de adicionar, e execute o programa. A que corresponde o resultado obtido? Confirme o que a função *fetchone*() faz, digitando *help(cur)* no interpretador do Python.

4. A seguinte linha de código permite obter uma contagem das linhas obtidas/afetadas após a execução de uma consulta (ou atualização/remoção):

```
print(cur.rowcount)
```

Altere agora a consulta SQL por forma a conseguir imprimir uma contagem dos empregados de apelido 'Silva'.

5. Uma exceção à extração de valores a partir de um tuplo acontece quando o tuplo é constituído por um elemento apenas. Na verdade, neste caso, o tuplo é constituído por um valor seguido de uma vírgula. Simule este cenário executando uma consulta que devolve uma coluna apenas (*nemp*):

```
import psycopg2

conn = psycopg2.connect("host=localhost dbname=postgres user=postgres password=postgres")
cur = conn.cursor()

cur.execute("SELECT nemp FROM emp")

primeira_linha = cur.fetchone()
print(primeira_linha)            # imprime (Decimal('1839'),)
val, = primeira_linha            # extrai o valor corretamente para a variável val!
print(val)                       # imprime 1839

cur.close()
conn.close()
```

Note a vírgula colocada depois da variável *val*!
Modifique agora a *query* para *SELECT nemp FROM emp WHERE nemp=9* e execute. Neste caso, a consulta não devolve linhas pelo que quando tenta imprimir *primeira_linha* obtém *None*. Quando na linha seguinte tenta extrair o resultado (vazio) da consulta para a variável val obtém um erro. Para não obter este erro na linha de código seguinte, pode:

a) Remover a vírgula (no entanto ficará com o valor *None* na variável *val*);
b) Impedir a extração do valor utilizando um *if* que verifica quantas linhas devolveu a consulta.

Experimente a solução a) e implemente de seguida a b).

6. Em alternativa ao desempacotamento de cada linha resultante de uma consulta em variáveis, é possível também aceder aos diferentes valores em cada linha especificando os nomes das respetivas colunas. Para isto, o cursor é criado com a seguinte configuração adicional:

```
import psycopg2.extras

...

cur = conn.cursor(cursor_factory=psycopg2.extras.DictCursor)
```

Experimente agora consultar nome e local de todos os departamentos especificando o nome da respetiva coluna de interesse, da seguinte forma:

```
for linha in cur.fetchall():
    x1 = linha['nome']
    x2 = linha['local']
    print(x1, x2)
```

7. O seguinte código insere um novo departamento na base de dados. Note a utilização da função *commit()* que permite terminar a transação, tornando definitivas as alterações à base de dados.

```
import psycopg2

conn = psycopg2.connect("host=localhost dbname=postgres user=postgres password=postgres")
cur = conn.cursor()

# Faz uma inserção na base de dados
cur.execute("INSERT INTO dep (ndep, nome, local) VALUES (60, 'Marketing', 'Faro')")

# Importante! Torna as alterações à base de dados persistentes
conn.commit()

cur.close()
conn.close()
```

Note a utilização de aspas e apóstrofos no comando INSERT. Execute o código e verifique no *PgAdmin* a correta inserção do departamento novo.

Tente criar agora:

a) Um programa que consiga apagar o departamento 60 da base de dados.
b) Um programa que peça um número N e um nome ao utilizador e permita atualizar o nome do departamento de número igual a N, com o nome inserido pelo utilizador. O seu programa deve imprimir a nova instrução SQL antes de a executar (para efeitos de verificação visual, apenas). Certifique-se ainda que o utilizador insere efetivamente um número antes de efetuar o acesso à base de dados.

8. O seguinte exemplo de código cria e executa um *prepared statement*, que consulta toda a informação dos empregados cujo salário está dentro de um certo intervalo de valores.

```
min = 100000
max = 200000
cur.execute("select * from emp where sal > %s and sal < %s", (min, max))
```

a. Use um *prepared statement* para mostrar toda a informação do empregado 1839.
b. Crie o código que, num primeiro passo, peça um conjunto de identificadores de empregados ao utilizador. Note que para parar o processo de inserção de dados deve ser introduzido o valor "0". De seguida deve executar um *prepared statement* que mostre todas as diferentes funções (i.e., sem repetição dos respetivos empregados).